

Optimizing Your Pipelines for Business Concerns

IMPLEMENTING NOTIFICATIONS IN AZURE PIPELINES



Chris B. Behrens

SOFTWARE ARCHITECT

@chrisbbehrens



Bad News Matters More Than Good News



That's the happy path



Knowing when things
are broken is more
important



We're doing an entire
course on this subject



Easy Notification in Azure DevOps



Azure DevOps has built-in
notifications



NO CODING
(not even scripting)



Capacity Planning for Continuing Builds

Planning for parallel builds

Planning for parallel *jobs*



Is This Actually a Problem?



My first piece of advice about a long-running build...

- Don't sweat it unless it's causing problems

If it is, however...

- Fixing the problem begins with understanding *why*
- And *why* begins with understanding what the build is doing



A Nightmare Solution



Story time



The compilation of the project was taking WAY too long



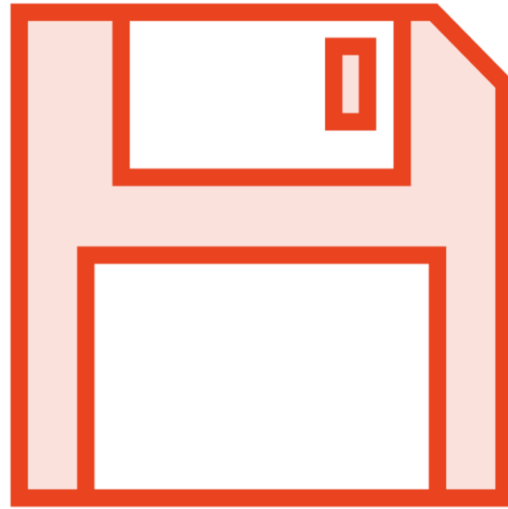
Every dependency was being built at web application compile-time



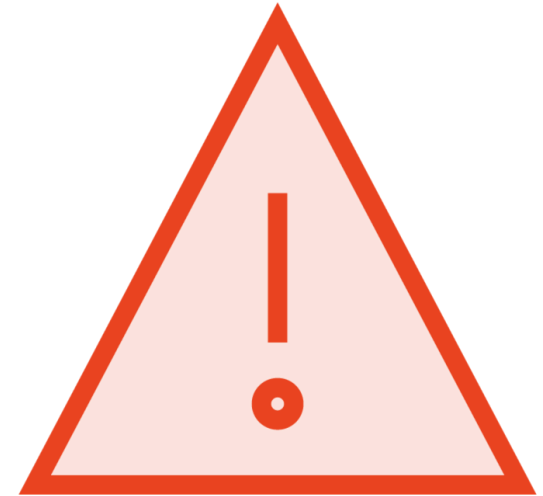
Thinking through Your Dependencies



“What else can we do?”



Some of them hadn't changed in years



And if they did, it should be a big deal



- A. You're only spending build time when something has changed
- B. That time may be out of sync with your primary build time
- C. Your references are versioned, so even if a new package breaks, the effects should be limited

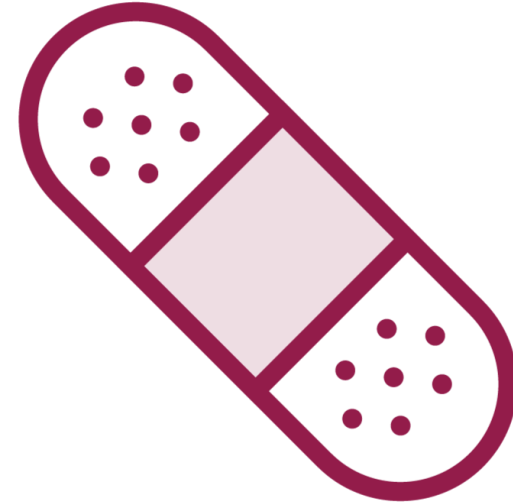
1. Split the solution
2. Create a new artifact feed
3. Build and publish the package to the new feed
4. Reference the package in the feed



A Tough Road



This is a long, hard road for a development team



Another short-term approach while you figure out your dependencies

Build Parallelism

1. Build Project A
2. Build Project B which depends on Project A
3. Build Project C which depends on Project B
4. ...and so on up until the top of the stack
5. Build the Web Application that depends on this entire stack



Build Parallelism

1. Build Project A
 2. Build Project B which depends on Project A
 3. Build Project C which depends on Project B
 4. ...
-
1. Build Project Email_A
 2. Build Project Email_B
 3. ...
-
1. Build Project PDF Exporter A
 2. Build Project PDF Exporter B
 3. ...
-
1. Build the Web Application that depends on those stacks



Build Parallelism

**The maximum of the build
times of the stack**



**The time to build the web
application**



How Many Parallel Jobs?



How many?

At first glance, three

But this is only true if they're all roughly the same size

Because they weren't, I could split the build in only two pieces

Otherwise, the fastest agent would just finish early and sit idle



Estimate that you'll need
one parallel job for every
four to five users in your
organization.



Our Build Is Compiling Quickly Now

**This is valid, but
complex**

**All the
compilation issues
are resolved**

Minutes, not hours



Splitting a Test Suite with Parallel Jobs



Nevertheless, the pipeline is taking too long to complete



The problem is in the testing phase



The compilation cannot be parallelized (at least not in this way)



But your tests should be parallelizable



If not all of them, then most of them

